

# תקציר:

המשחק הוא משחק יריות בדו מימד. במשחק השחקנים יצטרכו להתקדם בשלבי המשחק. בכל שלב יהיו אויבים או חידות שהשחקנים יצטרכו לפתור כדי להתקדם הלאה במשחק.

כל שחקן יוכל לסחוב עליו 8 חפצים שונים שכוללים נשקים ושיקויי חיים ומפתחות לפתיחת דלתות להתקדמות הלאה במשחק.

לכל שחקן יהיה נשק בסיסי שאיתו הוא יתחיל, במהלך המשחק האויבים יפילו חפצים ונשקים אשר כל שחקן יוכל לאסוף. וגם יהיו תיבות אוצר שהשחקנים יוכלו לפתוח ולהשיג חפצים.

היחודיות של המשחק היא מערכת הלחימה במשחק. מערכת הלחימה במשחק מאפשרת להילחם באויבים בצורה יחודית בכך שאפשר להילחם באויבים בדו מימד ולכוון לכל הכיוונים ולא כמו ברוב משחקי היריות בדו מימד, שאפשר לכוון בעיקר לצדדים.

המשחק מיועד להיות משחק עם שלט של קונסולה מכיוון שיותר כיף לכוון ככה על האויבים אבל ניתן לשחק גם עם מקלדת ועכבר.

בחירת שם המשחק:unboxinGrave מתאר חלק מהמשחק. במשחק יש קופסאות שמאוחריהן קברים, ויש פה משחק מילים unboxing grave  
במהלך המשחק יש קופסאות שאותן ניתן להרוס ומאחורי חלק מהן מופיעים קברים. מהקברים יוצאים אויבים שתוקפים את השחקנים.

# מנוע המשחק: MONOGAME

המשחק מפותח בתוכנה שנקראית monogame , תוכנה זו בנויה היא בעצם framework (בעברית: מסגרת ) שכוללת בתוכה בסיס לפיתוח משחק.

ב2004 מייקרוסופט הוציאה כלי לפיתוח משחקים שנקרא xna , ב2009 מונוגיים יצאה לשוק בחינם והיא הייתה בנויה על xna של מייקרוסופט ושדרגה את הפלטפורמה שמייקרוסופט יצרה.

# תוכן עניינים

[תקציר: 1](#_Toc64464727)

[מנוע המשחק: MONOGAME 1](#_Toc64464728)

[תוכן עניינים 2](#_Toc64464729)

[כפתורים במשחק 3](#_Toc64464730)

[חלוקת המסך 3](#_Toc64464731)

[שלבים במשחק: 3](#_Toc64464732)

[קברים: 3](#_Toc64464733)

[קופסאות: 3](#_Toc64464734)

[נשקים 4](#_Toc64464735)

[אויבים 4](#_Toc64464736)

[החלטות עיצוביות במשחק: 5](#_Toc64464737)

[מציאת מסלול: 5](#_Toc64464738)

[יצירת שלבים: 6](#_Toc64464739)

[עיצוב המפה: 10](#_Toc64464740)

[הסבר על השרת: 12](#_Toc64464741)

[הרעיון לפיתוח: 14](#_Toc64464742)

[הקוד:המחלקות במשחק 15](#_Toc64464743)

[GameClient: 16](#_Toc64464744)

# כפתורים במשחק

את המשחק ישחקו עם שלטים של קונסולה או באמצעות מקלדת ועכבר.

# חלוקת המסך

כל שחקן יראה במסך שלו את רשימת החפצים שלו (8 חפצים לכל שחקן).

בכל ריבוע של חפצים יהיה ניתן לשים כמה חפצים מאותו סוג. נגיד 5 עצים בריבוע אחד.

מעל לכל דמות יהיה health bar שיראה כמה חיים נשארו לשחקן.

בצד שמאל למעלה מופיע כפתור ההגדרות המאפשר לחזור לתפריט הראשי ולהגדיל/להקטין את המסך

# שלבים במשחק:

כל שלב במשחק יכיל מפה מיוחדת לשלב. כדי לעבור שלב השחקן יצטרך לזוז לקצה של המפה ובכך הוא יעבור לשלב הבא.

כל שלב יכיל חידות/פאזלים/אויבים שהשחקן יצטרך לעבור. כמו כן יהיו תיבות אוצר שהשחקן יוכל לפתוח וקופסאות שהוא יוכל להרוס.

בנוסף יהיו קברים במשחק שמהן יצאו אויבים.

# קברים:

הקברים במשחק יחולקו לשני סוגים: קברים שמזמנים אויב בטעינת השלב, וקברים שמזמנים אויב כאשר השחקן מתקרב לקבר

# קופסאות:

קופסאות יופיעו בסביבת השחקן, מאחורי הקופסאות יוכלו להופיע דברים נוספים שיתגלו רק שהשחקן יהרוס את הקופסאות, למשל יוכל להופיע קבר שייזמן אויב ברגע שהשחקן יגלה את הקבר או להופיע תיבת אוצר שתתן לשחקן חפצים וכסף.

# נשקים

כל הנשקים יהיו עם אינסוף תחמושת.

סוגי הנשקים:

צלף, m16, pistol, עוזי, מכונת יריה

**הצלף:**דוגמה של צלף בדו מימד:



יהיה קו שיוצא מהדמות שמראה לאן השחקן מכוון.

**הpistol** יהיה נשק בסיסי שעבור כל לחיצה על כפתור הוא ירה בכיוון היריה וכל השחקנים יתחילו איתו.

**הm16** יהיה אוטומטי ועבור לחיצה רצופה הוא יירה ברצף לעבר הכיוון הרצוי.

**העוזי** גם יהיה אוטומטי אבל הוא לא יהיה מדויק ויירה לאיזור הרלוונטי.

**המכונת יריה** תהיה לטווח קרוב יותר ותעשה הרבה dmg.

# אויבים

סוגי אויבים:

צייד: הצייד ינוע לעבר קבוצת השחקנים וינסה לתקוף אותם או את הבניין הקרוב ביותר אליו בדרך. הוא ינוע במהירות גבוהה לעבר השחקנים וינסה לתקוף אותם מקרוב

# החלטות עיצוביות במשחק:

במהלך הכנת המשחק נתקלתי בדילמות והייתי צריך לחשוב על הפתרון הטוב ביותר לפתור אותן.

# מציאת מסלול:

במהלך פיתוח המשחק יצרתי אויבים שתוקפים את השחקן, בהתחלה גרמתי לאויבים ללכת בקו ישר לעבר השחקן. לאחר מכן החלטתי להוסיף מפות למשחק ונתקלתי בבעיה, האויבים נעו דרך הקירות לשחקן. הבעיה הזו היא בעיה מוכרת במשחקי מחשב והיא נקראת pathfinding בעברית: מציאת מסלול.

בעיית מציאת המסלול היא בעיה שניתן לפתור אותה בכל מיני דרכים, הדרך שבה אני בחרתי לפתור אותה היא באמצעות שימוש באלגוריתמים שמחשבים מסלול קצר ביותר בגרף. גרף הוא בעצם נקודות(שנקראות קודקודים) שמחוברות זו לזו באמצעות קשתות. במדעי המחשב קיימים אלגוריתמים שמחשבים מסלולים שעוברים דרך הקשתות הללו בגרפים.

לכן, כדי לגשת לבעיה הכנתי מפה שבה קבעתי מיקום לשחקן ומיקום לאויב ושמתי קיר באמצע ביניהם. רציתי לייצר מסלול לאויב שיגיע לשחקן שלא עובר דרך הקיר.כדי לפתור את הבעיה הייתי צריך להמיר את הבעיה לבעיית גרפים.

המרת הבעיה לגרף: המפה של המשחק מורכבת מריבועים. כל ריבוע או שניתן לעבור בו או שלא ניתן לעבור בו מכיוון שיש בו קיר. לכן ניתן כבר להמיר את המפה לגרף: כל קודקוד בגרף מכיל ריבוע מהמסך שניתן לעבור בו או לא ניתן לעבור בו. המרת הנקודות במפה לגרף היא על ידי שימוש בפונקציה שממירה מיקום במפה לנקודה בגרף, וחיבור הנקודה לכל הנקודות שצמודות אליה, קודקוד שלא ניתן לעבור בו נסמן אותו כקודקוד שלא ניתן לעבור בו.

ניתן לגרף נקודת התחלה ונקודת סיום, נקודת ההתחלה תהיה האויב ונקודת הסיום תהיה השחקן. כעת ניתן להריץ אלגוריתם על הגרף שהתקבל שייחשב את המסלול מהאויב לשחקן.

בהמשך התגלתה לי בעייתיות בפתרון זה:

כל פעולת חיפוש מסלול היא מאוד איטית מכיוון שגודל המפה שלי הוא 1920\*1080 (גודל כל אריח הוא 16\*16) ולכן חיפוש מסלול מהאויב לשחקן יוכל לקחת זמן רב.

**הפתרון:** תחילה ניסיתי לפתור בעיה זו על ידי שימוש באלגוריתם A\* שהוא האלגוריתם שנחשב לרוב ליעיל ביותר למציאת מסלול קצר ביותר בגרפים. ואכן זמן מציאת המסלול השתפר.

לאחר מכן ניסיתי לבנות מפה שמכילה מבוך, וניתקלתי בבעיה חדשה, A\* אינו יעיל במקרה זה. A\* הוא אלגוריתם שמנסה לפתור את בעיית המסלול הקצר בכך שהוא "מכוון" את המסלול לכיוון נקודת הסיום. לכן כאשר יש מבוך עם פיתולים רבים אלגוריתם זה אינו יעיל, ושימוש באלגוריתם BreadthFirst יותר יעיל.

**הפתרון:** החלטתי שלאחר 1000 איטרציות של A\* אם לא נמצא מסלול, אז המסלול יחופש על ידי BreadthFirst ובכך האויב לא יחכה זמן רב מדי למסלול שלו.

לאחר מכן נתקלתי בבעיה נוספת: כאשר יש מספר גבוה של אויבים, המשחק נהיה איטי יותר מכיוון שהוא מחשב את כל המסלולים של כל האויבים לשחקן.

**הפתרון:** החלטתי להשתמש בthraed נוסף, thread במדעי המחשב הוא בעצם סוג של כלי במחשב שמאפשר למעבד לרוץ על כמה קטעי קוד במקביל. על ידי שימוש בthread נוסף יכולתי לחלק את עבודת המעבד ובכך להריץ ברקע את חישוב המסלולים של האויבים ולא להאט את המשחק.

# יצירת שלבים:

במהלך פיתוח המשחק רציתי ליצור מפה שהשחקן יוכל לנוע בא. תכננתי לעשות מפה שבנויה מtiles (אריחים), זו שיטה ליצירת מפות בדו מימד שבה המסך מחולק לריבועים קטנים ובכל ריבוע תופיע תמונה שתייצג את האובייקט במפה. למשל ניקח את התמונה הבאה:  


ניתן לראות שהתמונה מחולקת לריבועים. אפשר לספור כמה ריבועים כאלה יש: 8 על 15, ניתן לכל ריבוע מספר שייצג אותו. למשל עבור הריבוע בשורה 5 ועמודה 3 הוא יקבל : 5\*8 + 3 = 43. (כפול 8 מספר העמודות)

כעת ניתן באמצעות התוכנה monogame להשתמש בפונקציה שתבחר ריבוע ספציפי ותצייר אותו על המסך. באמצעות שיטה זו ניתן לייצר מפה על ידי מילוי המסך בריבועים כאלו על ידי התוכנה monogame.

**תהליך קבלת ההחלטות:** רציתי ליצור מפה (מפה היא הרקע שהשחקן רואה במהלך המשחק עצמו) למשחק, בחרתי להשתמש בשיטת האריחים.

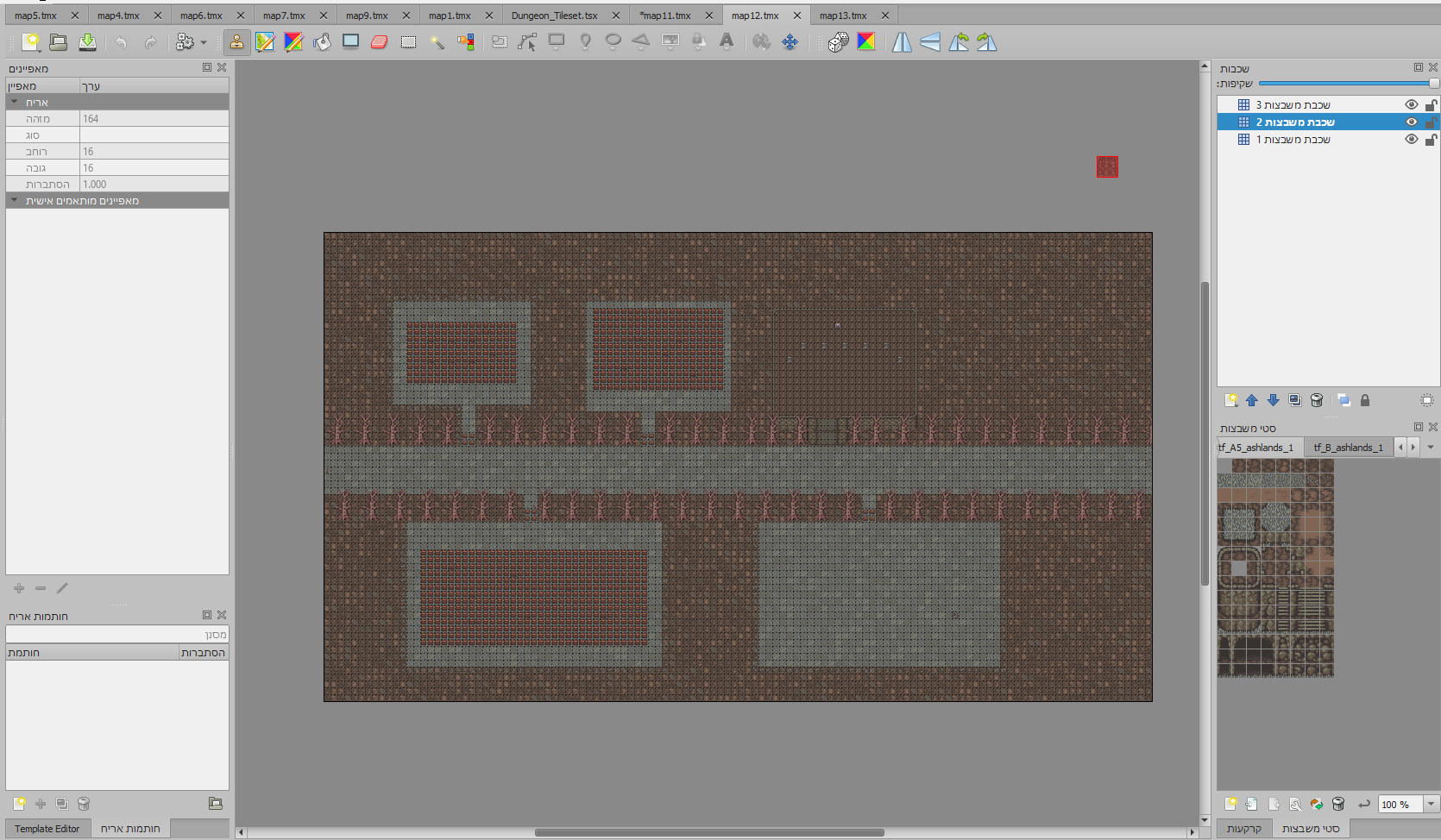
תחילה לא הייתי מודע לתהליך וכיצד יוצרים מפה על ידי אריחים. הרעיון שלי היה: אני איצור מערך דו מימדי שייצג את המסך. בכל מקום במערך אני אשים תמונה על ידי שימוש בפונקציה ממונוגיים. וכך תהיה לי מפה.

**הבעיתיות ברעיון זה: איך ניצור מפות?**  
כדי ליצור מפה, צריך לשים במערך בכל תא את המספר של התמונה, ואז לטעון את המפה על ידי פונקציית מונוגיים. לכן בשיטה זו אם הייתי רוצה ליצור מפה הייתי צריך לכתוב במערך כל תא ותא במפה.

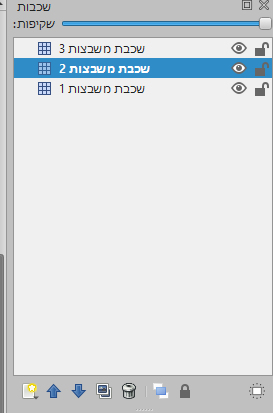
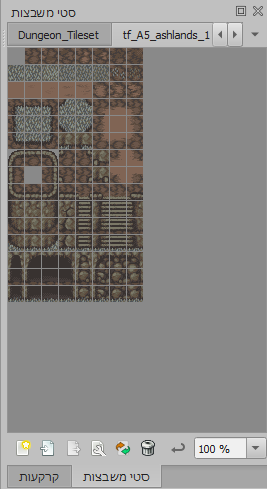
לאחר מכן חיפשתי פתרונות לבעיה והבנתי שיש כלים שעוזרים לך ליצור מפות. הדגש בפרוייקט היה לעשות משחק באופן עצמאי, וזו אחת מהנקודות היחידות שלא הצלחתי לחשוב על דרך אחרת ליצור מפה מבלי להשתמש בכלים שיעזרו לי, לכן החלטתי להשתמש בכלי ליצירת מפות.

הכלי שהשתמשתי בו ליצירת מפות: Tiled , תוכנה ליצירת מפות.

כך התוכנה נראית:



ניתן לראות שבמרכז המסך מופיעה המפה שאותה יוצרים. בצד ימין למעלה מופיעות השכבות המפה, ובצד שמאל למטה מופיעים האריחים שבתוכנה נקראים סט המשצבות:

שכבת המשבצות מייצגת שכבה של אריחים במפה. השכבה התחתונה מייצגת את הרצפה (שכבה שהשחקן נע עליה), השכבות העליונות יותר מייצגות קירות (השחקן לא יכול לנוע דרך הקירות).

איך התוכנה עובדת? יש לבחור גודל של מפה שאותו אנו רוצים ליצור. למשחק שלי השתמשתי בגודל 1920\*1080 שמייצג גודל סטנדרטי של מסך. בנוסף יש לבחור גודל אריח. אני בחרתי בגודל 16 על 16 (פיקסלים) . לאחר מכן יש לבחור תמונה של אריחים ולטעון אותה בתור סט משבצות. התוכנה תפרק את התמונה, ותיתן את האפשרות "לצייר" את המשבצות הללו על שכבת המשבצות.

לאחר יצירת המפה ניתן לייבא את קובץ המפה שיצרנו לתוכנה monogame. קובץ המפה מתלווה לשני סוגי קבצים נוספים שגם מייצגים את המפה.

סוגי הקבצים:

סיומת tmx – קובץ שמייצג את המפה עצמה, מכיל מידע על מיקומי הקבצים האחרים, מידע על איפה למקם על אריח במפה ובאיזו שכבה.

סיומת tsx – קובץ שמכיל מידע על התמונה שמייצגת את סט האריחים (המשבצות), הגובה והרוחב שלה והמיקום שלה והסוג שלה.

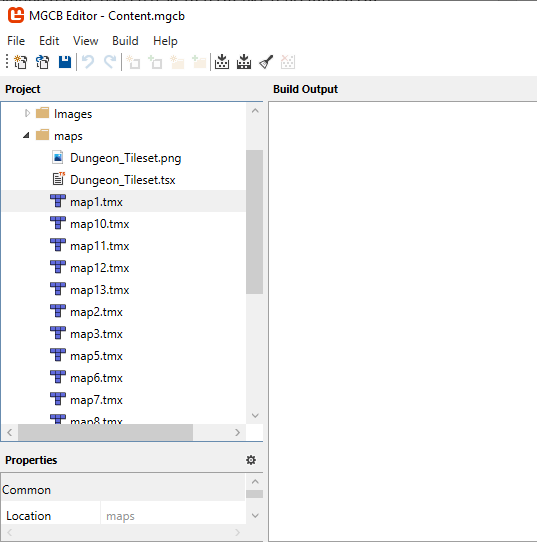
סיומת png – תמונה שמייצגת את המשבצות כמו התמונה הזו :



כעת שיצרנו מפה וקיבלנו 3 סוגי קבצים שונים, ניתן לטעון אותם לmonogame על ידי

שימוש במנהל התכנים של monogame.

ניתן לראות את הקבצים כך בתוכנה:



לאחר מכן ניתן ללחוץ על כפתור הbuild בתוכנה ובכך נוכל להשתמש בקבצים הללו בקוד שלנו.

המרת הקבצים למפות בקוד עצמו: כדי לטעון מפה למשחק עצמו, צריך להמיר את המפה לאובייקט בקוד, השתמשתי בתוסף לטעינת המפות שנקרא tiledSharp , ספרייה זו מקבלת קובץ של מפה והופכת אותו לאובייקט שמכיל את כל המידע הרלוונטי על המפה. לכן ניתן לתת לספרייה את הקובץ של המפה על ידי טעינתו ממנהל התכנים של מונוגיים. האובייקט של המפה מכיל שכבות של המפה, שמייצגות את השכבות שבהן השתמשנו בתוכנה Tiled .

כעת יש ברשותינו מערך שמייצג כל משבצת במפה, כל שנותר לנו לעשות על מנת להציג את המפה על המסך, זה לעבור על המפה על ידי שימוש בלולאות ושימוש בפונקציה Draw שמציירת על המסך במונוגיים, ואז המפה תוצג על המסך.

# עיצוב המפה:

כעת בשלב זה בפרוייקט הצלחתי ליצור מפה ולטעון אותה למסך, כעת הייתי צריך לחשוב איך למקם אובייקטים במפה. אובייקטים במפה אלו אובייקטים שיכולים להיות כל דבר שהשחקן יוצר איתם אינטרקציה. האובייקטים הראשוניים שעלו לי לראש היו קברים, כאשר שחקן מתקרב לקבר אז יוצא משם אויב, קופסאות שהשחקן יכול להרוס, תיבות אוצר שהשחקן יכול לפתוח.

לכן הייתי צריך דרך לסמן במפה איפה יש אובייקטים מיוחדים. הדרך שבה בחרתי היא דרך התוכנה Tiled , בחרתי פשוט לשים אובייקט במפה בנקודה מסויימת. כאשר אני אטען את המפה בנקודה זו, אני אטען את האובייקט הרלוונטי בקוד ויציב אותו בנקודה זו. יכולתי לדעת איזה אובייקט מפה לטעון לפי מספר האריח. (ציינתי מקודם שלכל אריח יש מספר שמחושב לפי העמודה והשורה שלו).

# הסבר על השרת:

**הסבר על הserver (שרת):**

יצרתי שרת שבנוי על monogame, השרת מתחבר ללקוח באמצעות sockets ובאמצעות הפקודות: beginAccept() beginReceive() send() הוא מתחבר ללקוחות ושולח להם מידע.

המידע מועבר באמצעות packets (פקטות)

הפקטות מורכבות מclass אבא שנקרא packetStructure שבו יש buffer ועליו נדפיס את הנתונים שאנו נרצה להעביר בפקט. קיימות מחלקות שיורשות מpacketStructure

שהן מימושים שונים של פקטות שונות שברצוננו להעביר מהשרת ללקוח ולהפך.

בחרתי לחלק את הפקטות לshort ולlong כאשר short יהיה פקט שיישלח הרבה ויכיל מידע קריטי על מיקומי אובייקטים. ויהיה פקט long שיכיל מידע פחות קריטי וכללי יותר.

כל הפונקציות שקשורות לרשת יימצאו במחלקה networkManager.

**תהליך התחברות של לקוח חדש לשרת:**

נניח שהרצנו את השרת, עכשיו השרת מריץ את הפקודה beginAccept ומחכה להתחברות הלקוח.

נריץ את הלקוח.

הלקוח רץ ומריץ את הפקודה beginConnect להתחברות לשרת. השרת מקבל את החיבור ומוסיף את הלקוח.

הוספת הלקוח בשרת תבוצע כך: השרת יוצר אובייקט player שהוא מייצג את הלקוח שהתחבר. כל המידע שרלוונטי ללקוח נמצא באובייקט זה.

לאחר שהשרת הוסיף את האובייקט player ללקוח הוא יתחיל לשלוח לו הודעות עם מידע על כל האובייקטים הרלוונטים במשחק.

**המחלקה HandlePacket:**

המחלקה handlePacket היא מחלקה שתטפל בהודעות שיגיעו לשרת/לקוח.

בשרת לכל לקוח קיים handlePacket שיטפל בהודעות שלו.

**תהליך טיפול בהודעה של handlePacket בלקוח:**

הגיע הודעה מהשרת והלקוח יקרא לפקודה handle שנמצאת במחלקה handlePacket

כדי למנוע מצב שבו התקבלו 2 הודעות ונצטרך לטפל ב2 הודעות באותו זמן ובכך יווצרו בעיות רבות, הלקוח מאפשר טיפול רק בהודעה אחת בו זמנית. הלקוח משתמש בפקודה:

Interlocked.Exchange ובכך רק thread אחד יוכל לטפל בהודעה בכל זמן שהוא.

הטיפול בהודעות עצמן יתבצע דרך הפקודה update כדי למנוע בעיות.(בעיות של שימוש באותם משאבים באותו זמן).

תהליך הטיפול בהודעה של handlePacket בשרת יהיה זהה פרט לכך שבשרת עבור כל לקוח יהיה handlePacket שונה.

**מבנה ההודעות:**

ההודעות שישלחו יורכבו מ2 ביטים ראשונים: גודל ההודעה, ו2 הביטים הבאים יהיו סוג ההודעה.

סוגי ההודעות הן:

1 - short packet from client to server

2 - short packet from server to client

3 - long packet from server to client

4 - long packet from client to server

**הערה חשובה על השרת:**

השרת יעביר את המידע עם פקטות שיכילו רק מידע מאוד מצומצם כדי לא להעמיס על הרשת. לכן השימוש בcollectionManager חשוב. השרת ישלח מספרים שייצגו חפצים ואויבים במקום לשלוח את כל המידע שצריך על האויבים והחפצים. לדוגמה המספר 1 ייצג את האויב "שלד" שיש לו מהירות מסוימת, חיים מסויימים וטסקטורה ועוד דברים. אבל השרת רק יצטרך לשלוח את המספר שמייצג את השלד. מה שכן השרת יצטרך לעדכן כל שחקן עם המיקום של השלד ע"י שליחת וקטור מיקום של השלד וגם לעדכן את החיים של השלד בכל רגע.

באופן דומה ישלחו פקטות שייצגו נשקים חפצים ועוד.

במוד שחקן יחיד הgame\_client יבצע את כל מה שצריך כדי להפעיל את המשחק.

במוד מרובה משתתפים השרת יבצע חלק גדול מהפעולות שצריכות להיות משותפות לכולם. הפעולות שהשרת יבצע:

ניהול האויבים: כאשר שחקן יפגע באויב ישלח פקט לשרת שבו מידע על הנזק שהשחקן גרם לאויב. השרת יחשב את כמות החיים שנשאר לאויב ובשניה שהחיים של האויב יגיעו ל0 השרת ישלח פקט לכל השחקנים שהאויב מת.

באופן דומה השרת ינהל את החיים של כל שחקן.

# הרעיון לפיתוח:

**תפריט ראשי:**

בתפריט הראשי יהיה אפשרות לבחור אם לשחק single player או לשחק multiplayer

בנוסף יהיה כפתור how to play , כפתור high scores וכפתור exit

**מה אני הולך לפתח במשחק:**

אני אתחיל לפתח את התשתית למשחק שתורכב כדלקמן:

המחלקה client תריץ את המחלקה game\_client שהיא בעצם המחלקה שהיא המשחק עצמו.

בתוך game\_client יאותחל תחילה התפריט הראשי שימומש במחלקה משלו בשם MainManager

לאחר שהמשתמש יבחר לשחק הgame\_client יצא מהתפריט הראשי ויפעיל את המשחק.

המשחק עצמו ינוהל לפי שלבים, ע"י מחלקה level\_manager שתחליט איזה שלב להפעיל.

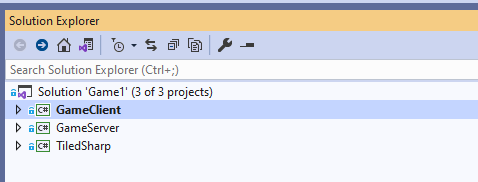
**שלבים:**

אני הולך ליצור שלבים למשחק, כל שלב יכיל בתוכו חידה/אויבים/פאזל שהשחקנים יצטרכו לעבור. כל שלב אני אכין בתוכנה שנקראית Tiled , בתוכנה ניתן ליצור מפה/רקע למשחקי דו מימד. המפה תכיל כל מיני אובייקטים מיוחדים: קופסאות, תיבות, קברים וכו'. אני אטען את המפה על ידי שימוש בתוסף TiledSharp שיכול לקרוא את המפות שנוצרות בTiled ולהמיר אותן לקוד. אני אקרא את המפה באמצעות פונקציה ואשמור אותה במשתנה. לאחר מכן אני אעבור על המפה עם לולאה ואחפש כל אובייקט מיוחד. אם מצאתי אובייקט שהGID (gid זה מספר הזיהוי של אובייקטים במפה) שלו שווה לGID של אובייקט מיוחד אני אוסיף אובייקט מיוחד במפה במיקום זה.

**לדוגמה:** במפה עם תיבת אוצר, אני אקרא את המפה ואריץ לולאה שעוברת על כל תא במפה, אם מצאתי את התיבה אני אוסיף אובייקט למשחק שנקרא Chest במיקום של התיבה. השחקן יוכל לפתוח את התיבה מכיוון שהוספתי פונקציות מיוחדות לאובייקט זה.

# הקוד:המחלקות במשחק

**הפרוייקט עצמו נראה כך בתוכנה visual studio:**

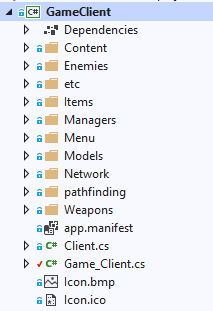


GameClient - המשחק עצמו, צד לקוח

GameServer - השרת של המשחק

TiledSharp – תוסף שהוספתי לפרוייקט, התוסף יכול לטעון מפות מהתוכנה Tiled שמייצרת מפות

# GameClient:



Client.cs – סקריפט שדרכו מורץ המשחק

Game\_Client.cs – המחלקה של המשחק עצמו, יורשת מהמחלקה game שמובנית בmonogame

**תיקיות:**

Content – תיקיה מובנת של monogame שמכילה את כל הcontent שנוסף למשחק.

Enemies – תיקיה שמכילה את כל המחלקות שקשורות לאויבים במשחק

Items – תיקיה שמכילה את כל המחלקות שקשורות לחפצים במשחק

Managers – תיקיה שמכילה את כל המחלקות שמנהלות את המשחק

Models – תיקיה שמכילה את כל המודלים במשחק

Menu – תיקיה שמכילה את כל המחלקות שקשורות לתפריט הראשי במשחק

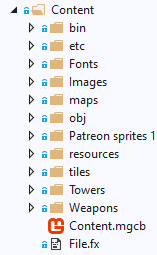
Network – תיקיה שמכילה את כל המחלקות שקשורות לחיבוריות לשרת במשחק

Pathfinding – תיקיה שמכילה את כל המחלקות שקשורות לpathfinding (מציאת מסלול של אויבים) במשחק

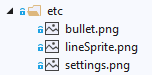
Weapons – תיקיה שמכילה את כל המחלקות שקשורות לנשקים במשחק

**תיקיות:**

**Content:**



bin – תיקיה מובנית של monogame

Etc - מכילה דברים שאינם דורשים תיקיה מיוחדת, תמונה של קו, תמונה של הגדרות. 

Fonts -תיקיה שמכילה פונטים למשחק

Images – תיקיה שמכילה תמונות של המשחק, רקע לתפריט הראשי, רקע להגדרות



Maps – תיקיה שמכילה מפות למשחק

Patreon sprites 1 – תיקיה שמכילה ספרייטים לדמויות במשחק

Resources – תיקיה שמכילה תמונות של החפצים במשחק

Weapons – תיקיה שמכילה תמונות של נשקים במשחק

**Enemies:**



Simple\_Enemy.cs – מחלקה שמייצגת אויב פשוט במשחק

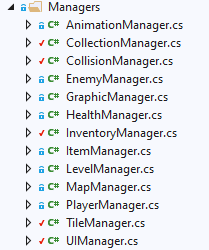
**Items:**



Item.cs – מחלקה שמייצגת חפץ במשחק.

ItemStock.cs – מחלקה שמייצגת חפץ בinventory במשחק. כל חפץ יכול להיות מוחזק בinvetory בכמות מקסימלית בכל ריבוע של inventory, במחלקה הזו יש חפץ ואת הכמות שלו בinventory בריבוע מסויים בinventory

**Managers:**



AnimationManager – מחלקה שמנהלת אנימציה מסויימת, בהינתן קבוצת אנימציות כלשהי המחלקה תנהל את האנימציות ותחליט איזו אנימציה להפעיל.

CollectionManager – מחלקה שבה כל התוכן של המשחק נוצר בטעינת המשחק. בין התוכן שנוצר – נשקים, סוגי תחמושות, אויבים, חפצים, אנימציות. ניתן לקרוא לפונקציה בcollectionManager שתחזיר העתק של אובייקט(שטענו בתחילת המשחק).

CollisionManager – מחלקה שאחראית על בדיקת התנגשויות של אובייקטים אחד בשני.

EnemyManager – מחלקה שאחראית על כל האויבים במשחק

GraphicManager – מחלקה שאחראית על פונקציות גראפיות במשחק כמו טעינת אנימציה, חישוב ריבועים של אנימציה, שינוי מסך למסך מלא.

HealthManager – מחלקה שאחראית להציג את מד החיים של כל אויב ושחקן במשחק.

InventoryManger – מחלקה שאחראית על מערכת אחזקת החפצים במשחק

ItemManager – מחלקה שאחראית על פונקציות שקשורות לחפצים במשחק, הפלת חפץ לרצפה, מציאת החפץ הקרוב ביותר ועוד.

LevelManager – מחלקה שאחראית על ניהול השלבים במשחק.

MapManager – מחלקה שאחראית על כל אובייקטי המפה המיוחדים: קופסאות, תיבות, דלתות.

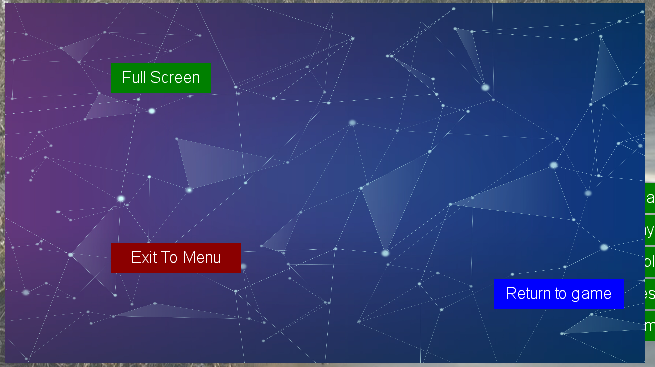
PlayerManager – מחלקה שאחראית על כל השחקנים במשחק, מחזיקה מערך עם כל השחקנים במשחק וניתן להוסיף בה שחקנים.

TileManager – מחלקה שאחראית על טעינת המפה, ציור המפה בכל קריאה לפונקציה Draw.

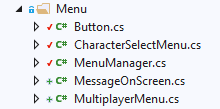
UIManger – מחלקה שאחראית על שכבת הUI בתוך המשחק, מכילה חלון של הגדרות, תצוגת הכסף במשחק. מציגה כפתור שמופיע בצד שמאל במסך (בכל שלב במשחק) לפתיחת ההגדרות:



חלון ההגדרות נראה כך:



**Menu:**



Button – אובייקט שמייצג כפתור בתפריטים במשחק



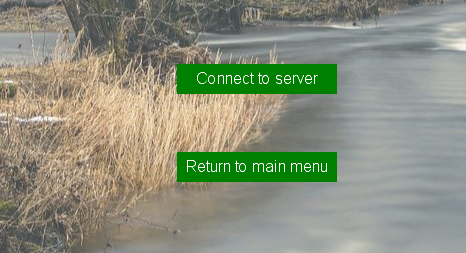
CharacterSelectMenu – תפריט שבו ניתן לבחור דמות ולהתחיל את המשחק



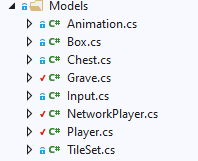
MenuManager – מחלקה שמנהלת את כל התפריטים במשחק וגם מכילה את התפריט הראשי:



MultiPlayerMenu – חלון שמציג אפשרות להתחבר לשרת, לאחר ההתחברות השחקן מועבר לחלון בחירת דמות וניתן להכנס למשחק



**Models:**



Animation – אובייקט שמייצג אנימציה, מכיל רשימה של תמונות שמייצגות אנימציה

Box – אובייקט במפה שמייצג קופסה שניתן להרוס.

Chest - אובייקט במפה שמייצג תיבת אוצר שניתן לפתוח ולקבל חפצים וכסף

Grave – אובייקט במפה שמזמן אויב כאשר מתקיימים תנאים מסויימים כמו מרחק מהשחקן ואם האובייקט לא מאחורי קופסא

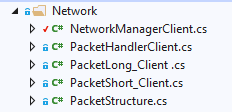
Input – מחלקה שמייצגת קלט, לאובייקט של השחקן יהיה קלט מקשים מסויים שייחודי לו.

NetworkPlayer – מחלקה שמייצגת שחקן מהרשת, דומה למחלקה player רק שלא מתבצעים כמעט שום חישובים במחלקה זו מכיוון שהסרבר שולח מידע על שחקנים אחרים ואין צורך לבצע פעולות מיוחדות במחלקה זו, רק להציג את השחקנים האחרים ברשת.

Player – אובייקט שמייצג את השחקן, מבוצעות בו הפעולות של קריאת הקלט מהמשתמש, עדכון אנימציה של השחקן, עדכון החיים של השחקן.

TileSet – אובייקט שמכיל טקסטורה של מפה. משומש בTileManager

**Network:**



networkManagerClient – מחלקה המשמשת להתחברות לשרת ולביצוע כל פעולות שליחה וקבלה של מידע מהשרת

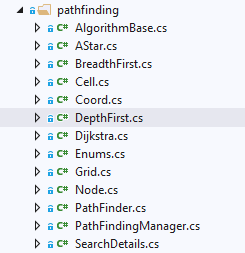
PacketHandlerClient – מחלקה המנהלת את קבלת ההודעות מהשרת

PacketLong\_client – מחלקה שיורשת מpacketStructure ומייצגת הודעה שמכילה מידע שנשלח לשרת לעיתים רחוקות

PacketShort\_client – מחלקה שיורשת מpacketStructure ומייצגת הודעה שמכילה מידע שנשלח לשרת לעיתים קרובות

PacketStructure – אובייקט שמייצג הודעה שתישלח לשרת. מכיל buffer שמייצג את ההודעה שניתן להוסיף לו מידע.

**Pathfinding:**



algorihmBase – מחלקת בסיס של אלגוריתם מציאת דרך

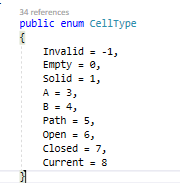
Astar – מחלקה שמכילה אלגוריתם A\* שיורשת מalgorithmbase

BreadthFirst – מחלקה שמכילה אלגוריתם חיפוש BFS שיורשת מalgorithmbase

Cell – אובייקט שמייצג תא בגרף לצורך האלגוריתם חיפוש. יכול להכיל תא ריק או תא מלא שלא ניתן לעבור דרכו בחיפוש מסלול.

Coord – מייצג נקודה בגרף, משומש בתוך Cell

Enums – מחלקה שמכילה enum עם סוגי תאים בגרף



Grid – מייצג גרף שמורכב מרשימה של cell.

Node – מייצג איבר במסלול של האלגוריתם למציאת מסלול.

PathFinder – אובייקט שניתן לכל אויב במפה. מחשב מסלול מהאויב לשחקן על ידי שימוש בגרף שמיוצר בTileManager בטעינת מפה, והפעלת אלגוריתמים עליו של חיפוש מסלול.

PathFindingManager – מכיל רשימה של כל הPathFinder. בהיווצרות המחלקה נוצר thread שבו רצה לולאה שקוראת בכל איטרציה לחישוב מסלול pathfinder אחר. הלולאה רצה על כל הpathfinder אין סוף פעמים ותמיד מחשבת מחדש כל מסלול של כל אויב

SearchDetails – אובייקט שמייצג תוצאות של חישוב מסלול, הוא מכיל את המסלול שנמצא בין האויב לשחקן או שהוא מכיל משתנה שאומר שאין מסלול בין האויב לשחקן.